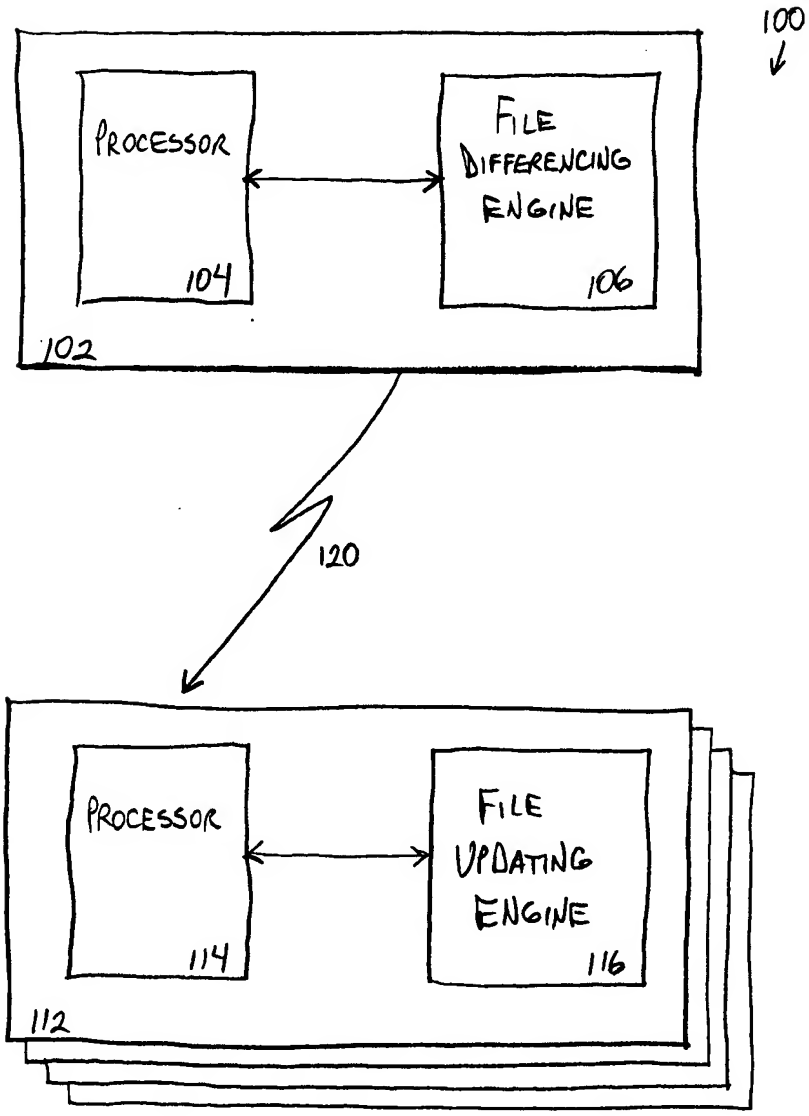
100
PROCESSOR
104
FILE
DIFFERENCING
ENGINE
106
102
120
PROCESSOR
114
FILE
UPDATING
ENGINE
116
112

FIGURE 1

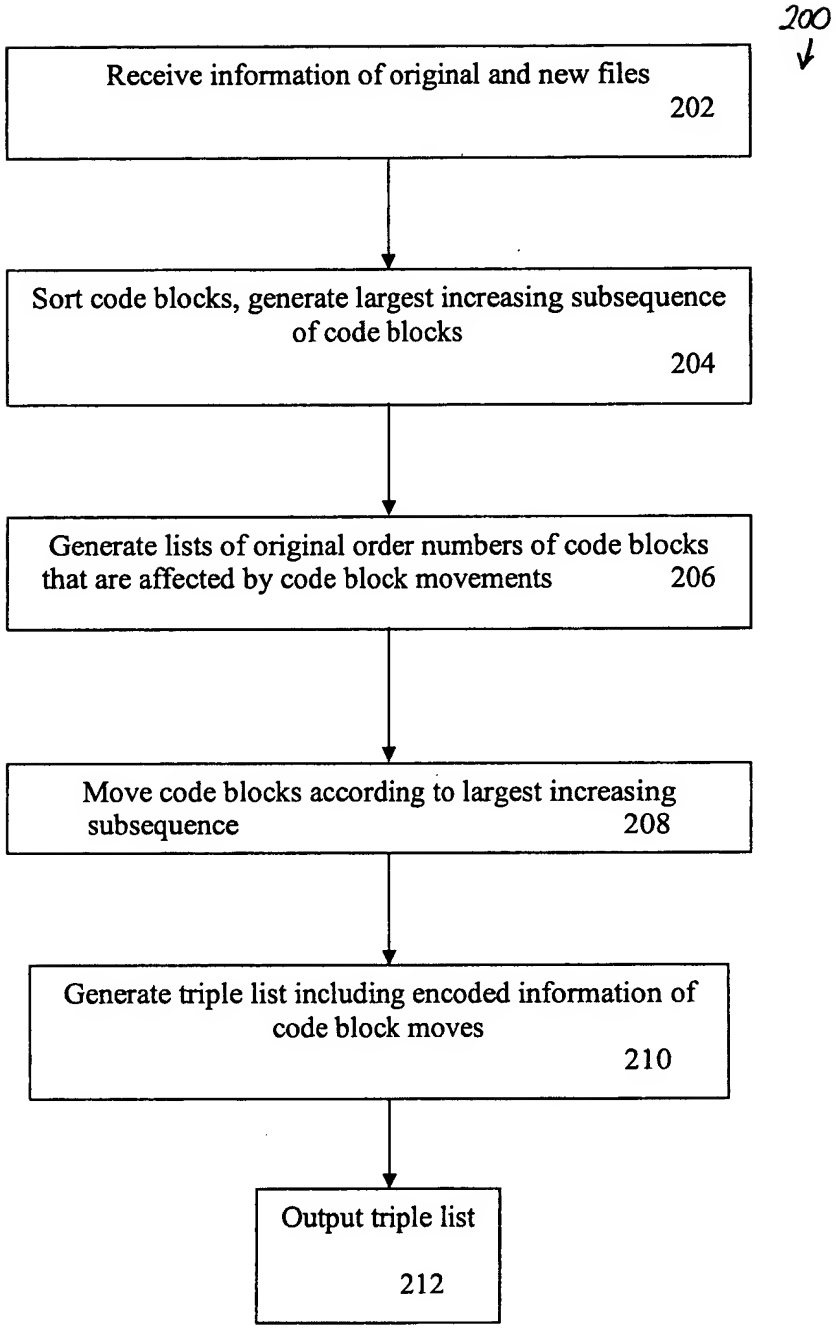
200
Receive information of original and new files
202
Sort code blocks, generate largest increasing subsequence of code blocks
204
Generate lists of original order numbers of code blocks that are affected by code block movements
206
Move code blocks according to largest increasing subsequence
208
Generate triple list including encoded information of code block moves
210
Output triple list
212

Figure 2

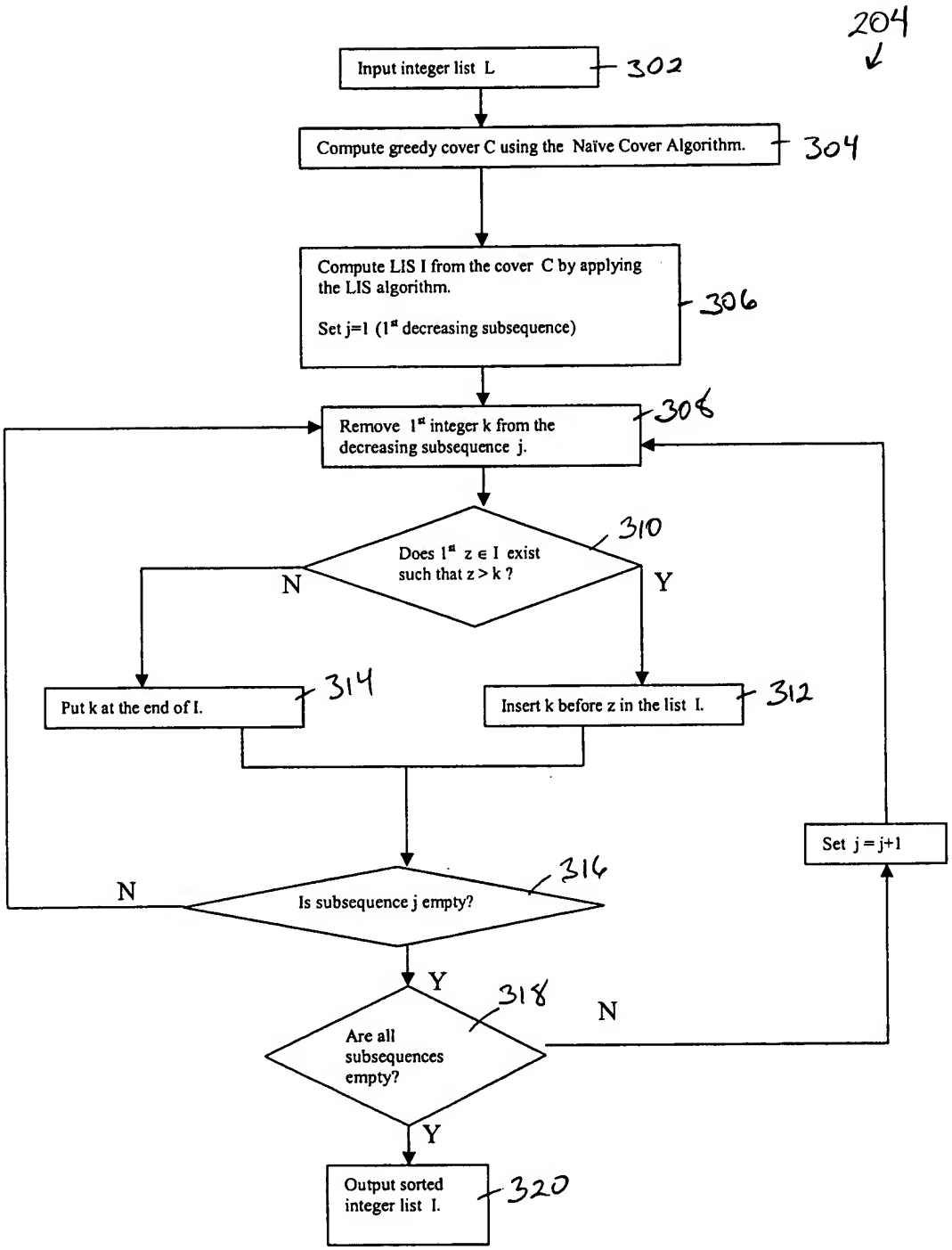
204
Input integer list L
302
Compute greedy cover C using the Naïve Cover Algorithm.
304
Compute LIS I from the cover C by applying the LIS algorithm.
Set j=1 (1st decreasing subsequence)
306
Remove 1st integer k from the decreasing subsequence j.
308
Does 1st z ∈ I exist such that z > k ?
310
N
Y
Put k at the end of I.
314
Insert k before z in the list I.
312
Set j = j+1
Is subsequence j empty?
316
N
Y
Are all subsequences empty?
318
N
Y
Output sorted integer list I.
320

Figure 3

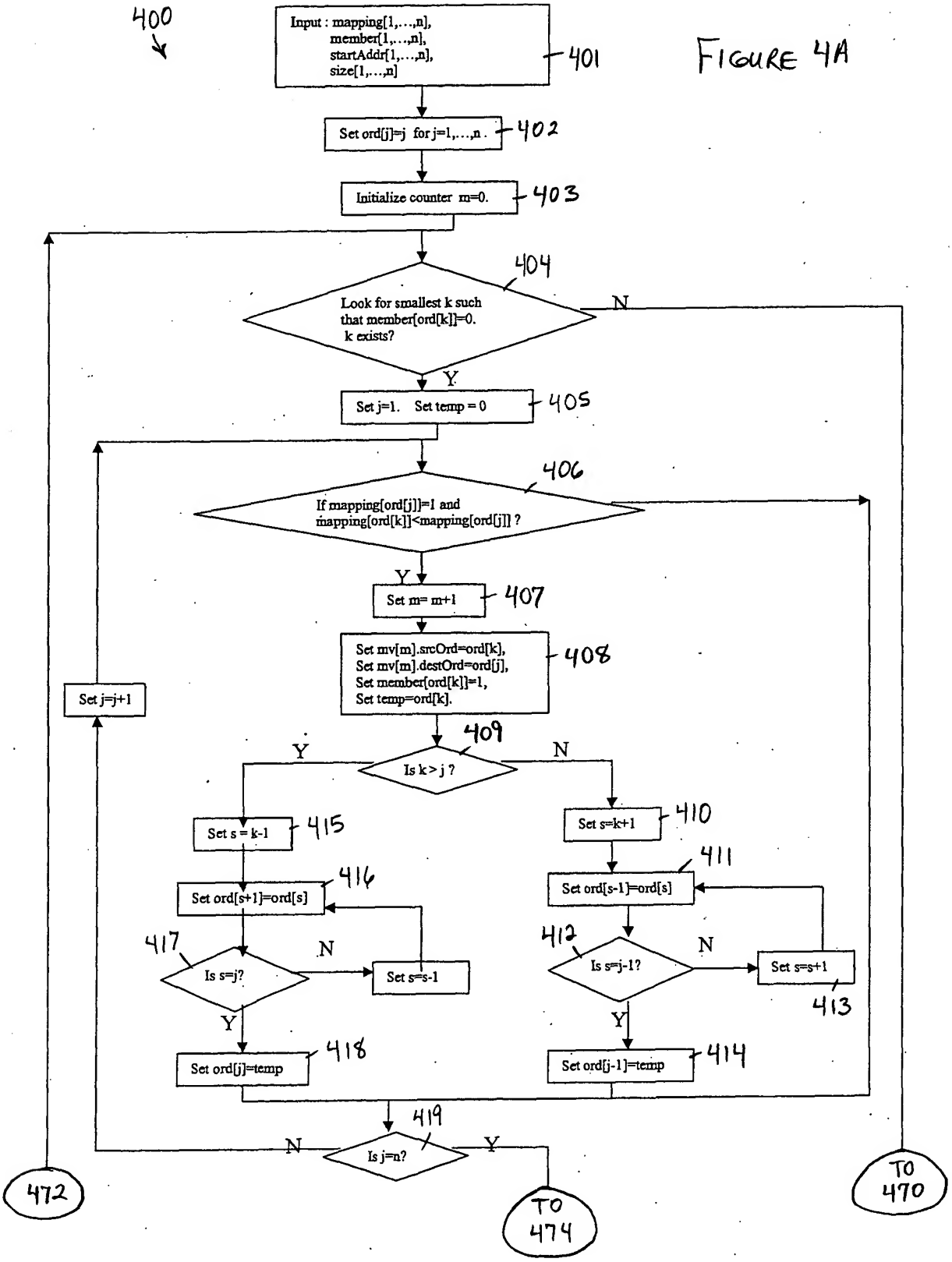
400
Input : mapping[1,...,n],
member[1,...,n],
startAddr[1,...,n],
size[1,...,n]
401
Set ord[j]=j for j=1,...,n.
402
Initialize counter m=0.
403
404
Look for smallest k such that member[ord[k]]=0. k exists?
N
Y
Set j=1. Set temp = 0
405
406
If mapping[ord[j]]=1 and mapping[ord[k]]<mapping[ord[j]] ?
Y
Set m= m+1
407
Set mv[m].srcOrd=ord[k],
Set mv[m].destOrd=ord[j],
Set member[ord[k]]=1,
Set temp=ord[k].
408
Set j=j+1
409
Y
Is k > j ?
N
Set s = k-1
415
Set s=k+1
410
411
Set ord[s-1]=ord[s]
416
Set ord[s+1]=ord[s]
417
Is s=j?
N
Set s=s-1
412
Is s=j-1?
N
Set s=s+1
413
Y
Y
Set ord[j]=temp
418
Set ord[j-1]=temp
414
419
Is j=n?
N
Y
472
TO 474
TO 470

FIGURE 4A

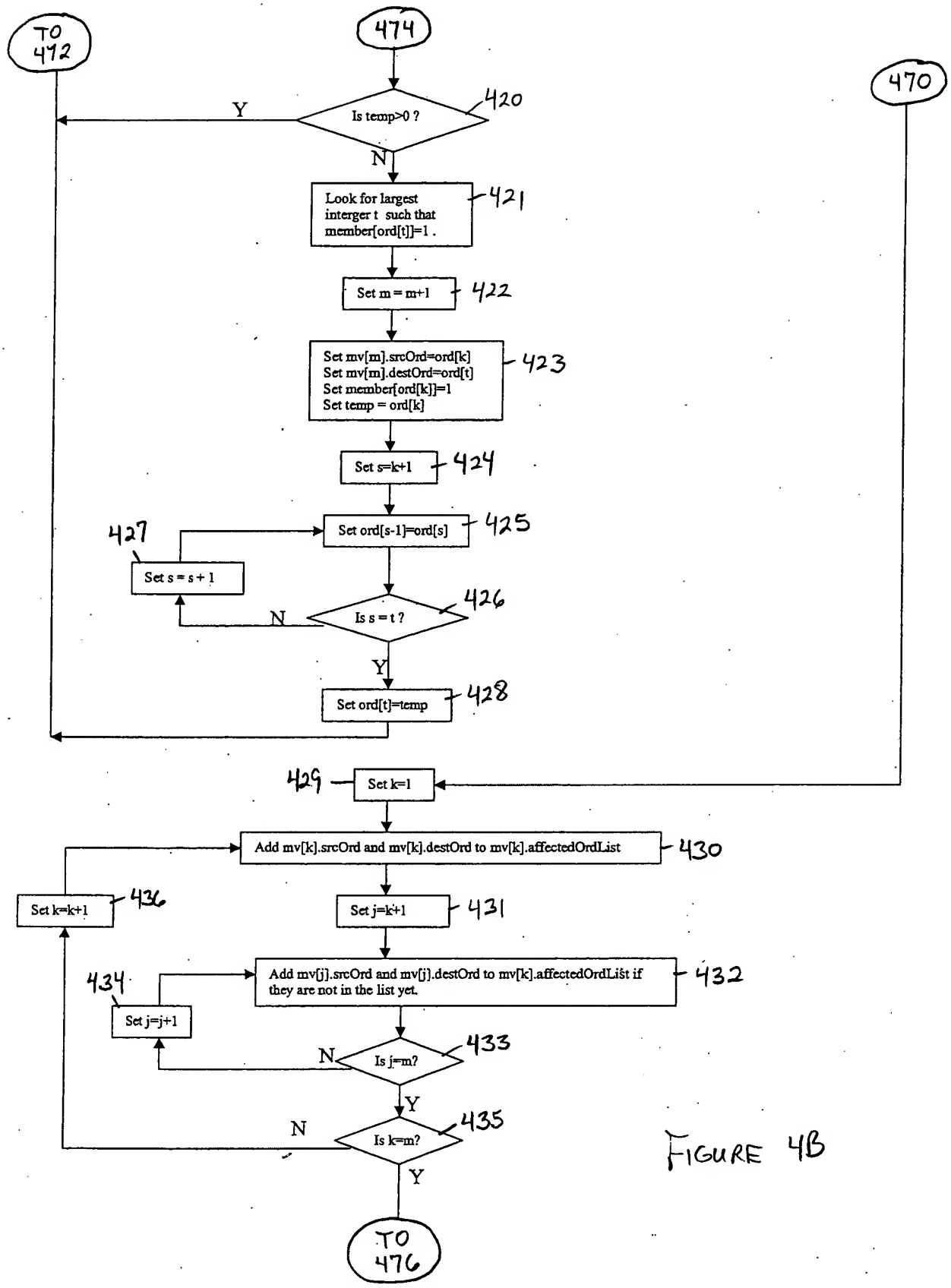
TO 472
474
470
Is temp>0 ?
420
Y
N
Look for largest interger t such that member[ord[t]]=1 .
421
Set m = m+1
422
Set mv[m].srcOrd=ord[k]
Set mv[m].destOrd=ord[t]
Set member[ord[k]]=1
Set temp = ord[k]
423
Set s=k+1
424
Set ord[s-1]=ord[s]
425
427
Set s = s + 1
Is s = t ?
426
N
Y
Set ord[t]=temp
428
429
Set k=1
Add mv[k].srcOrd and mv[k].destOrd to mv[k].affectedOrdList
430
Set k=k+1
436
Set j=k+1
431
434
Add mv[j].srcOrd and mv[j].destOrd to mv[k].affectedOrdList if they are not in the list yet.
432
Set j=j+1
Is j=m?
433
N
Y
Is k=m?
435
N
Y
TO 476

FIGURE 4B

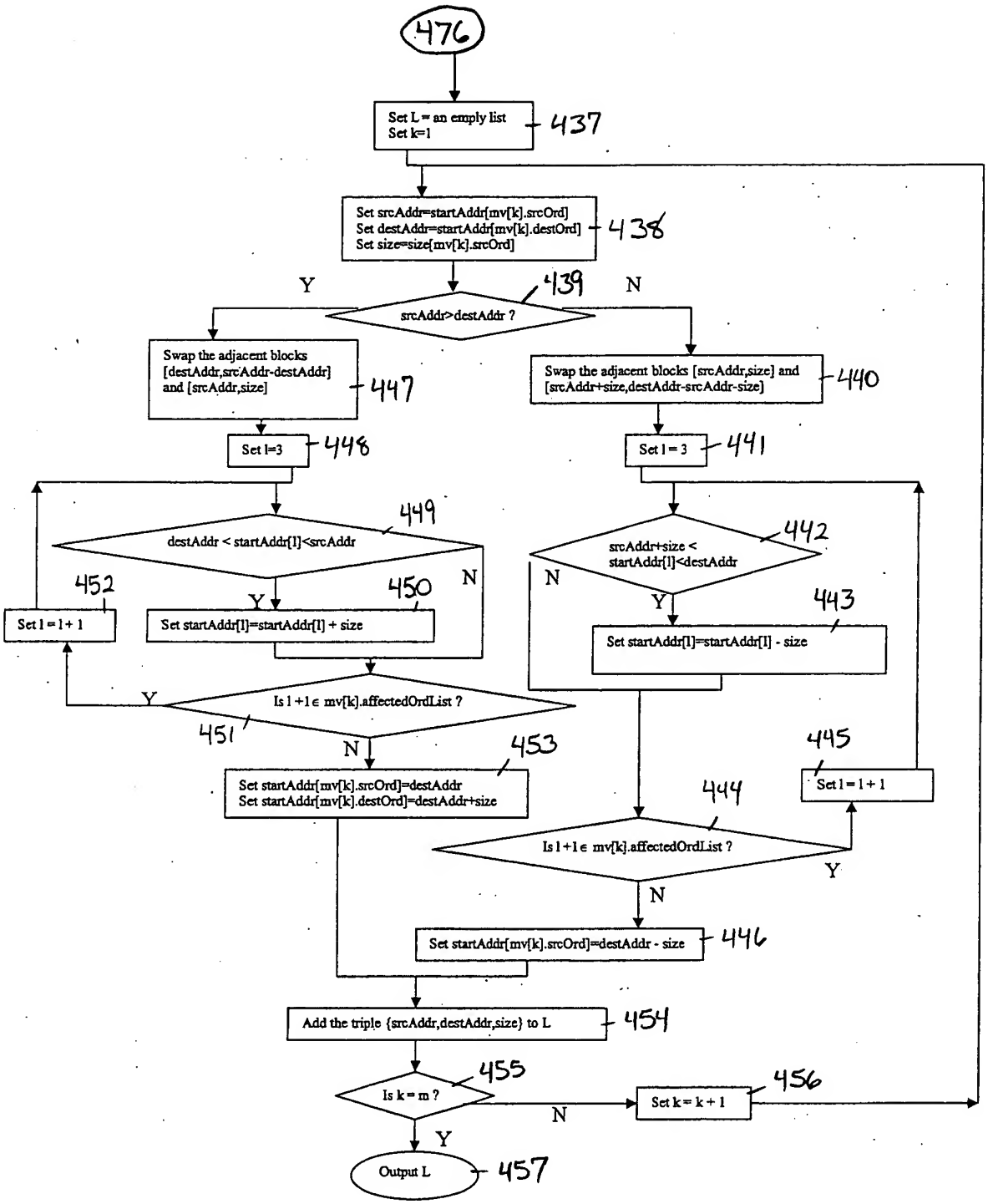
476
Set L = an empty list
Set k=1
437
Set srcAddr=startAddr[mv[k].srcOrd]
Set destAddr=startAddr[mv[k].destOrd]
Set size=size[mv[k].srcOrd]
438
Y
439
N
srcAddr>destAddr ?
Swap the adjacent blocks [destAddr,srcAddr-destAddr] and [srcAddr,size]
447
Swap the adjacent blocks [srcAddr,size] and [srcAddr+size,destAddr-srcAddr-size]
440
Set l=3
448
Set l = 3
441
449
destAddr < startAddr[l]<srcAddr
442
srcAddr+size < startAddr[l]<destAddr
N
Y
452
Set l = l + 1
Y
450
Set startAddr[l]=startAddr[l] + size
N
N
Y
443
Set startAddr[l]=startAddr[l] - size
Y
Is l +1 ∈ mv[k].affectedOrdList ?
451
N
453
Set startAddr[mv[k].srcOrd]=destAddr
Set startAddr[mv[k].destOrd]=destAddr+size
445
Set l = l + 1
444
Is l +1 ∈ mv[k].affectedOrdList ?
Y
N
Set startAddr[mv[k].srcOrd]=destAddr - size
446
Add the triple {srcAddr,destAddr,size} to L
454
455
Is k = m ?
N
Set k = k + 1
456
Y
Output L
457

FIGURE 4C

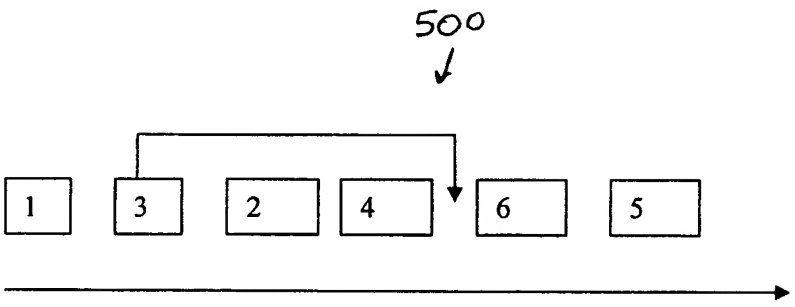

Figure 5

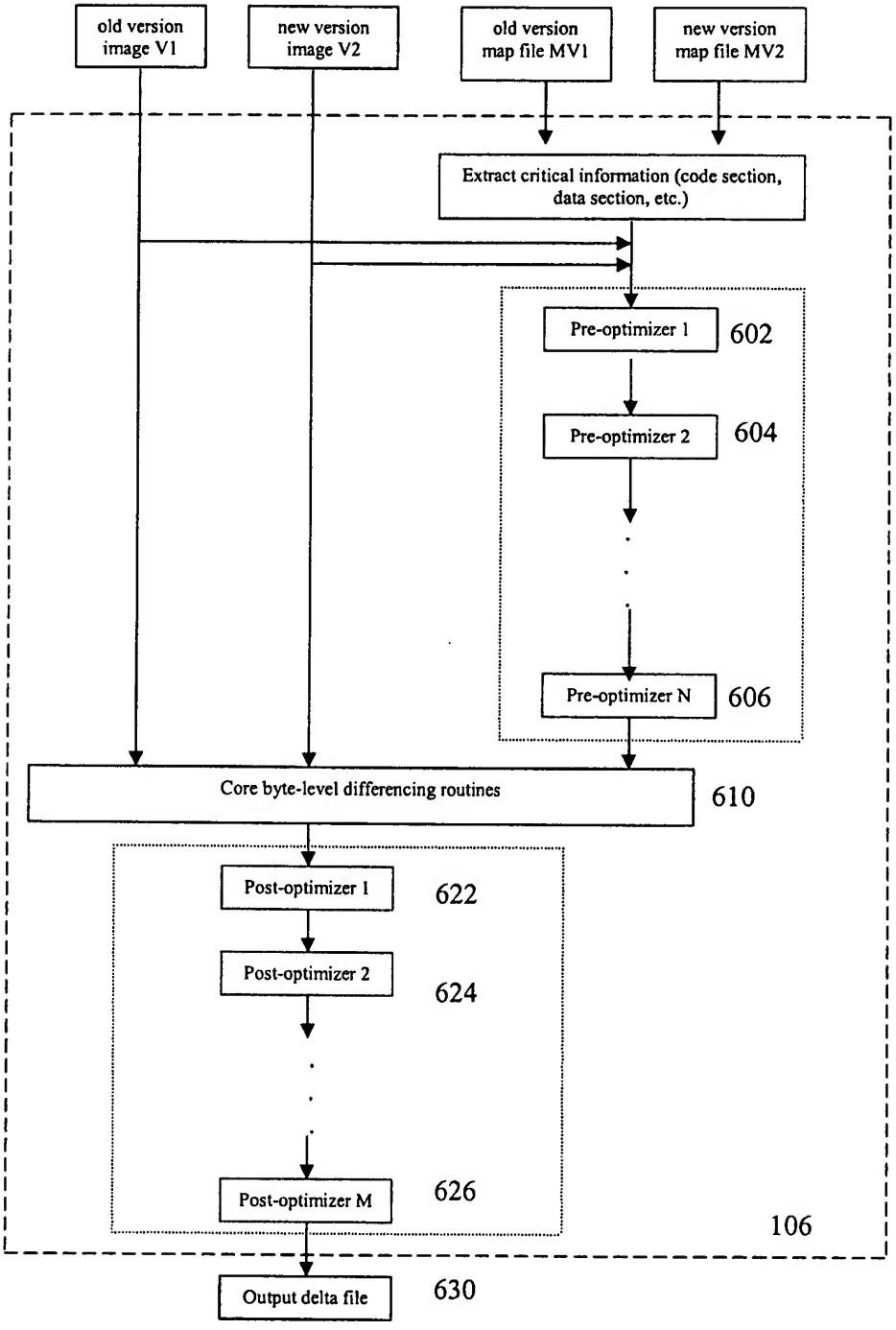
old version image V1
new version image V2
old version map file MV1
new version map file MV2
Extract critical information (code section, data section, etc.)
Pre-optimizer 1
602
Pre-optimizer 2
604
Pre-optimizer N
606
Core byte-level differencing routines
610
Post-optimizer 1
622
Post-optimizer 2
624
Post-optimizer M
626
106
Output delta file
630

Figure 6